REMARKS

This paper is responsive to the Office Action mailed May 7, 2007. Claims 1-63 are pending. Claims 1-63 have been rejected.

At paragraph 3 of the Office Action, the Examiner requires cross-reference to related applications. Applicant amends the specification accordingly.

The Examiner further suggests arranging the specification as provided in 37 CFR 1.77(b). As the Examiner correctly points out, the language of 37 CFR 1.77(b) uses the term "should" (e.g., "The specification should include the following sections in order", emphasis added). The Applicant therefore submits that this arrangement is suggested but not required. Further, 37 CFR 1.77(c) states, "The text of the specification section defined in paragraphs (b)(1) through (b)(12) of this section, if applicable, should be preceded by a section heading in uppercase and without underlining or bold type." (emphasis added). The Applicant therefore submits that 37 CFR 1.77 suggests using these section headings only if they are applicable. Accordingly, the Applicant respectfully requests that the Examiner accept the specification without modifying the arrangement.

At paragraph 5 of the Office Action, the Examiner rejects claims 1, 22 and 43 under 35 U.S.C. 102(b) as being anticipated by "ALTO: A Platform for Object Code Modification; 1999 (Muth). The Applicant traverses these rejections. The Applicant submits that Muth does not teach or suggest all of the limitations of independent claims 1, 22 and 43.

Muth does not describe "a method of grouping subject code during a translation of subject code into translated target code to account for self-modifying subject code" (emphasis added). The sections of Muth at pages 140 and 40 specifically referred to by the Examiner do not relate to grouping of subject code to account for self-modifying code.

At page 140, "partitions" are mentioned in relation to basic blocks in the context of "Procedural Abstraction" as set out in Section 6.3 of Muth. At page 140, Muth describes a process for identifying groups of basic blocks which together define a set of basic blocks having single-

entry and single-exit. When identified, as such Muth teaches the use of the procedural abstraction techniques on the functions of the groups of basic blocks, not just on individual basic blocks. Such grouping does not take account of self-modifying code.

Page 40 of Muth describes how Muth deals with self modifying code. Muth teaches that some self-modification events cannot be handled correctly (page 40, lines 5-6). Muth teaches that the ALTO program is not applied to self modifying code events, and that the self-modifying code is allowed to operate without ALTO 's intervention, being treated by ALTO as an unknown function (page 40, lines 12-14). Muth does not describe "identifying self-modifying code events in said subject code during translation of subject code into translated code and also during subsequent execution of translated code." The example in Figure 2(a) featuring a self-modifying code event is explicitly stated as an example of why self-modifying code is not handled in ALTO (see page 43, lines 6-13).

The passages of Muth at page 41 and 135 are irrelevant to the step of "dividing a region of memory containing said subject code into at least one subject instruction group of subject addresses when identifying a self-modifying code event, wherein each subject instruction group includes one or more ranges of subject code addresses in said memory which are affected by a respective self-modifying code event." Page 41 describes the process of address translation which is performed by the ALTO program when it re-writes a binary, and in these passages there is no mention of the action being taken when identifying a self-modifying code event in the subject code. As mentioned above, the example in Figure 2(a) of Muth, which features a self-modifying code event, will not work in ALTO (see page 43, lines 6-13).

Page 135 describes grouping basic blocks into partitions of similar blocks as a step in producing identical basic blocks. This has nothing to do with self-modifying code events.

Since Muth fails to teach or suggest all of the limitations of claims 1, 22 and 43, those claims should be allowable. Since claims 2-21 depend from allowable base claim 1, claims 23-42

depend from allowable base claim 22, and claims 44-63 depend from allowable base claim 43, those claims should also be allowable.

At paragraph 6, the Examiner rejects claims 1-63 under 35 U.S.C. 102(b) as being anticipated by "A Robust Foundation for Binary Translation of X86 Code," 1987 (Hsu). The Applicant traverses this rejection. The Applicant submits that Hsu does not teach or suggest all of the limitations of independent claims 1, 22 and 43.

In particular, Hsu does not teach or suggest "a method of grouping subject code during a translation of subject code into translated target code to account for self-modifying subject code, comprising identifying self-modifying code events in said subject code during translation of subject code into translated code and also during subsequent execution of translated code." The passage referred to on page 3 of Hsu makes it clear that although self-modifying code is detected, the detection of self-modifying code is performed "to prevent the binary translator from translating self-modifying code" (page 3, lines 16-17). Thus in Hsu, there is no grouping of subject code during a translation of subject code to account for the self-modifying code, there is no link between the identified self-modifying code and the translation, nor is there a link between the self modifying code and the subsequent execution of translated code.

The step of "dividing a region of memory containing said subject code into at least one subject instruction group of subject addresses when identifying a self-modifying code event, wherein each subject instruction group includes one or more ranges of subject code addresses in said memory which are affected by a respective self-modifying code event" is not disclosed in Hsu. Hsu discloses the merging of adjacent instruction areas, but not with reference to areas of memory which are affected by self-modifying code events. Chapter 6 (pages 55-65) of Hsu describes how Hsu deals with self-modifying code, where it is clear that the translator does not form groups of subject instructions which are affected by self-modifying code events. Page 64, lines 7-10 of Hsu make it clear that self-modifying code is treated as a special case different from the general operation of the translator, in that self-modifying instructions are dealt with individually by single-stepping through them.

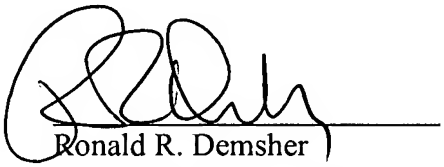
Since Hsu fails to teach or suggest all of the limitations of claims 1, 22 and 43, those claims should be allowable. Since claims 2-21 depend from allowable base claim 1, claims 23-42 depend from allowable base claim 22, and claims 44-63 depend from allowable base claim 43, those claims should also be allowable.

Filed herewith is a Request for a One-Month Extension of Time, which extends the statutory period for response to expire on September 7, 2007. Accordingly, Applicant respectfully submits that this response is being timely filed.

In view of the above amendment and remarks, the Applicant believes the pending application is in condition for allowance. No other fees are believed to be due in connection with the filing of this response, however the Commissioner is authorized to debit Deposit Account No. 08-0219 for any required fee necessary to maintain the pendency of this application.

Respectfully submitted,

Dated: September 7, 2007



Ronald R. Demsher
Registration No.: 42,478
Attorney for Applicant(s)

Wilmer Cutler Pickering Hale and Dorr LLP
60 State Street
Boston, Massachusetts 02109
(617) 526-6000 (telephone)
(617) 526-5000 (facsimile)